Smart Congestion Control for Delay- and Disruption Tolerant Networks

Aloizio P. Silva*, Katia Obraczka †, Scott Burleigh‡, Celso M. Hirata*

* Instituto Tecnológico de Aeronáutica, Computer Engineering Department, São Paulo, Brazil
† University of California Santa Cruz, Computer Engineering Department, California, USA
‡Jet Propulsion Laboratory - NASA/ California Institute of Technology, California, USA
{aloizio, hirata}@ita.br, katia@soe.ucsc.edu, scott.c.burleigh@jpl.nasa.gov

***Abstract*—In this paper, we propose a novel congestion control framework for delay- and disruption tolerant networks (DTNs). The proposed framework, called `Smart-DTN-CC`, adjusts its operation automatically as a function of the dynamics of the underlying network. It employs reinforcement learning, a machine learning technique known to be well suited to problems in which the *environment*, in this case the network, plays a crucial role; yet, no prior knowledge about the target environment can be assumed, i.e., the only way to acquire information about the environment is to interact with it through continuous online learning. Smart-DTN-CC nodes get input from the environment (e.g., its buffer occupancy, set of neighbors, etc), and, based on that information, choose an *action* to take from a set of possible actions. Depending on an action's effectiveness in controlling congestion, it will be given a *reward*. Smart-DTN-CC's goal is to maximize the overall reward which translates to minimizing congestion. To our knowledge, Smart-DTN-CC is the first DTN congestion control framework that has the ability to automatically and continuously adapt to the dynamics of the target environment which allows Smart-DTN-CC to deliver adequate performance in a variety of DTN applications and scenarios. As demonstrated by our experimental evaluation, Smart-DTN-CC is able to consistently outperform existing DTN congestion control mechanisms under a wide range of network conditions and characteristics.**

I. Introduction

Delay and Disruption Tolerant Networks (DTNs) are networks that operate in challenged- and extreme environments. Unlike traditional networks, such as the TCP/IP Internet, DTNs are often subject to high latency caused by very long propagation delays (e.g. interplanetary communication) and/or intermittent connectivity. As a result, in DTNs, there is no guarantee of continuous end-to-end connectivity between nodes. In fact, arbitrarily frequent and long-lived connectivity disruptions are part of DTNs' normal operation.

DTN's inability to guarantee end-to-end connectivity between nodes and extremely long latencies due to high propagation delays and/or episodic connectivity call for approaches to network control that are fundamentally different from what has been in use on the Internet. More specifically, the control functions provided by the Internet's Transmission Control Protocol (TCP) are all performed end-to-end over a logical connection established between sender and receiver, Since DTNs are typically "connectivity-challenged", network control in DTNs must be done on a hop-by-hop basis. This is the case for the custody transfer [1] and store-carry-and-forward [2] paradigms that have been proposed for DTNs which require that DTN nodes store data in persistent storage for arbitrarily long periods of time before they find a suitable next-hop. As a consequence, congestion control is critically important in order to ensure that DTN nodes are congestion-free such that they can serve as relays to help deliver messages end-to-end. DTN congestion control has thus received considerable attention from the networking research community. An overview of the current DTN congestion control state-of-the-art is presented in [3], while the performance of a representative set of DTN congestion control mechanisms is evaluated in [4] and [5]. These studies reveal that existing DTN congestion control solutions do not exhibit adequate performance when used in different scenarios and conditions. Existing DTN congestion control schemes do not perform adequately due mainly to their inability to dynamically adjust their operation to changing network conditions. Additionally, some of them employ reactive techniques, depend on the underlying routing mechanism, and use global network information.

Recently, computational intelligence has been successfully employed in a variety of applications ranging from robot control [6] [7] [8], routing protocol [9], data offloading [10], smart vehicles [11] [12], habitat and environmental monitoring [13][14], and medical diagnostics [15]. We propose a novel approach to DTN congestion control which uses computational intelligence techniques in order to automatically adapt to the dynamics of the underlying network without the need for human intervention. This is particularly important in remote and extreme environments. The proposed DTN congestion control framework, called `Smart-DTN-CC`, autonomically adapts the congestion control effort based on the dynamics- and operating conditions of the target environment. `Smart-DTN-CC` is based on reinforcement learning [16], a machine learning

technique which, given the current *environment*, takes *actions* with the goal of maximizing a cumulative *reward*. Reinforcement learning is based on the idea that if an action is followed by a satisfactory state (e.g., performance improvement), then the tendency to produce that action is strengthened (reinforced). On the other hand, if the state becomes unsatisfactory, then that particular action is penalized. Reinforcement learning is known to be well suited to problems in which the *environment* plays a paramount role but prior knowledge about it cannot be assumed either because it is not available or it is prohibitively expensive to collect. Therefore, the only (practical) way to acquire information about the environment is to learn by interacting with it "online".

In Smart-DTN-CC, a DTN node cycles through the following sequence of steps: (1) it gets input from the environment (e.g., its buffer occupancy, set of neighbors, etc); (2) uses that information as representative of the current state of the environment and, based on that knowledge, chooses an action to take from a set of possible actions; and (3) measures the reward resulting from the action taken. Smart-DTN-CC's overall goal is to maximize the reward, i.e., minimize node congestion. To our knowledge, Smart-DTN-CC is the first DTN congestion control framework that has the ability to automatically and continuously adapt to the dynamics of the target environment. This unique feature allows Smart-DTN-CC to deliver adequate performance in a variety of DTN applications and scenarios. As demonstrated by our experimental evaluation, Smart-DTN-CC is able to consistently outperform existing DTN congestion control mechanisms under a wide range of network conditions and characteristics.

The remainder of this paper is organized as follows. Section II introduces Smart-DTN-CC. Section III describes the experimental methodology we used to evaluate Smart-DTN-CC while Section IV presents our results. We conclude the paper with directions for future work in Section V.

II. Smart DTN Congestion Control

Computational intelligence techniques have been applied to a number of networking problems including TCP throughput prediction [17], network intrusion detection [18], loss classification to improve TCP congestion control in wireless networks [19], path quality prediction [20], content delivery networks (CDNs) [21], mobility prediction [22], round-trip time [23], and collision rate [24] estimation.

Smart-DTN-CC employs Q-Learning [25], a variant of reinforcement learning, in which the learning agent first uses online learning to build a model of the system and then utilizes this knowledge to find a satisfactory solution. We should also point out that one of our main design goals is to base congestion control decisions on local knowledge, e.g., information about the node itself and its neighbors. This is because in challenged network environments, obtaining global knowledge is either impractical or prohibitively expensive. However, if global information is available, Smart-DTN-CC can also use it as input.

A. Q-Learning

In Smart-DTN-CC, Q-learning builds a representation of the node's state in terms of *Q-values* and then uses these values to make congestion control decisions. Each node x in the network stores its own view of its state in a *Q-table*. In each position $Q(s, a)$ of its *Q-table*, where s are all possible states and a are the actions that are possible in each state, the node stores a *Q-value*. *Q-values* are estimates of the effectiveness of a node's different control strategy alternatives which are represented by the different actions nodes can take. The actions used in Smart-DTN-CC perform proactive or reactive congestion mitigation and are independent of the underlying routing protocol. Examples of Smart-DTN-CC actions include increase/decrease message generation rate, discard buffered messages (e.g., based on their age), broadcast congestion notification locally, and migrate messages to neighboring nodes (see Table I for a list of the actions that have been currently implemented).

Q-values are updated each time conditions (such as buffer occupancy, drop ratio, local congestion) change. The goal of the learning algorithm is to discover a policy that maximizes cumulative reward based on past experiences and interactions with the environment. Each node maintains a history represented by a sequence of *state-action-reward*,

$$\langle s_0, a_0, r_1 \rangle, \langle s_1, a_1, r_2 \rangle, \ldots \quad (1)$$

This means that the node was in state s_0, took action a_0, which resulted in it receiving reward r_1; the node transitioned to state s_1, took action a_1, and received reward r_2, and so on. At every iteration of our Q-learning algorithm, a node updates its *Q-table* based on its current state s and the selected action using Equation 3.

At each new experience the node is able to maintain an estimate of the Q^* function and adjusting *Q-values* based on actions taken and reward received. This is done using Sutton's prediction difference [26] which is the difference between the immediate reward received plus the discounted value (γ) of the next state and the *Q-values* of the current state-action pair (Equation 2).

$$r + \gamma V^*(s^{'}) - Q^*(s, a) \quad (2)$$

The reward function (see Section II-E) can produce different rewards each time the transition $\langle s, a \rangle$ is repeated. Consequently, the Q-learning algorithm will repeatedly alter the values of $Q^*(s, a)$ when the node receives a reward r. Where in each node's experience a decaying weighted average of the current Q^* values and the revised estimate are taken into account. Thus in Equation 3, Q_n^* denotes the estimate on the nth iteration of the algorithm.

$$Q_n^*(s,a) = (1-\eta_n)Q_{n-1}^*(s,a) + \eta_n(r + \gamma V_{n-1}^*(s^{'})) \quad (3)$$

where

$$\eta_n = \frac{1}{1 + visits_n(s,a)} \quad (4)$$

and

$$V_{n-1}^*(s^{'}) = max_{a^{'}}[Q_{n-1}^*(s^{'}, a^{'})] \quad (5)$$

Note that $visits_n(s,a)$ is the total number of times the state-action pair has been visited up to and including the nth iteration. Furthermore, the value of η_n in Equation 4 decreases as n increases, so that updates become smaller as training progresses. When reducing η_n during training, it is possible to reach the correct convergence of Q^* function. The value of $V_{n-1}^*(s^{'})$ represents the maximum reward that is attainable in the state following the current one, that is, it is an estimative of the optimal future value.

The standard Q-learning algorithm has several stages as shown in Algorithm 1. One particular benefit of using the Q-learning algorithm is that it does not require specific domain knowledge of the problem it is attempting to solve [27].

Algorithm 1 Q-learning algorithm

```
1: procedure Q-LEARNING
2:     For each s and a, initialize the table entry Q(s, a) to zero
3: top:
4:     Observe the current state s
5:     Select action a through an action selection method and execute it
6:     Receive reward r
7:     Observe the new state s' and update the table entry Q(s, a) using
       values of r and Q(s', a)
8:     s ← s'
9:     goto top.
10: end procedure
```

It is important to note that Q-learning does not specify what actions the node should take at each state as it updates its estimates. This means that Q-learning allows arbitrary experimentation while preserving the current best estimate. This is possible because Q-learning constructs a value function on the state-action, and since this function is updated according to the ostensibly optimal choice of action at the following state, it does not matter what action is being followed at that state. Because of this, the estimated returns in Q-learning are not contaminated by experimental actions [28]. Therefore Q-learning is not experimentation sensitive.

Q-learning allows nodes to use different action selection strategies. There are two main techniques for selecting action a from the possible actions in every state [29]: (1) **Exploration** allows selection of an action different from the one that is considered the current best and (2) **Exploitation** selects action a that maximizes $Q^*(s,a)$.

In the beginning of the learning process, action selection typically favors exploration whereas exploitation is preferred towards the end of learning. Section II-B describes two action selection strategies we implemented in Smart-DTN-CC and show their performance in Section IV.

B. Action Selection Methods

Our current implementation of Smart-DTN-CC employs two alternate action selection strategies which differ in how they combine action exploration and exploitation. Using the first action selection method we use, which was originally proposed in [28] and [30], nodes try out actions probabilistically based on their *Q-values* using a Boltzmann probability distribution. More specifically, given a state s, a node tries out action a with probability given by Equation 6.

$$\rho_s(a) = \frac{e^{\frac{Q^*(s,a)}{T}}}{\sum_{a' \in A} e^{\frac{Q^*(s,a')}{T}}} \quad (6)$$

Note that $e^{\frac{Q^*(s,a)}{T}} > 0$ whether *Q-values* are positive or negative. The "temperature" T controls the amount of exploration (the probability of executing actions other than the one with the highest *Q-values*). If T is high and *Q-values* are similar, an action will be chosen randomly. If T is low and *Q-values* are different, the action with the highest *Q-values* be selected.

The second action selection method, called WoLF for "Win or Learn Fast", was introduced in [31] and [32]. The basic idea is to use two learning rates "to maximize the probability of choosing a profitable action and slowly decrease the probability of an action that is less beneficial to the agent". The rationale behind the use of this strategy is that when a node is congested and it selects actions that return negative rewards, it must try to reduce the congestion level and make a slow transition to a less congested state. This means that the node should keep a reasonable probability of choosing actions that have proven to be profitable in the recent past. Hence, we want the node to gradually reduce its congestion level such that it keeps buffer utilization high.

WoLF defines two learning rates: γ_{min} and γ_{max}. They are used when the node is decreasing and increasing its rewards, respectively. Initially, because nodes are still exploring the environment, every action a has the same probability of being selected (Equation 7). Note that A denotes the set of all available actions for the node at state s.

$$\forall i \in A\colon \rho_{a_i} = \frac{1}{|A|} \quad (7)$$

Consequently, if the node took an action a_i at time step $t-1$ that increases its reward when comparing to the previous reward at time step t, at time t it updates the probability of selecting a_i in the future. The node performs this step using Equation 8, where $\rho_{a_i}^{t-1} + \gamma_{max} < 1$. In Equation 9, the probability of choosing another action j is randomly distributed [32].

$$\rho_{a_i}^t = \rho_{a_i}^{t-1} + \gamma_{max} \quad (8)$$

$$\rho_{a_j}^t = \frac{1-\rho_{a_i}^t}{|A-1|}, \forall j \in A, j \neq i \quad (9)$$

On the other hand, if the node's reward decreased, $\rho_{a_i}^t$ is updated as shown in Equation 10, where $\rho_{a_i}^{t-1} - \gamma_{min} > 0$. In Equation 11, the probability of choosing another action j is incremented at the same rate, with respect to its previous probability.

$$\rho_{a_i}^t = \rho_{a_i}^{t-1} - \gamma_{min} \quad (10)$$

$$\rho_{a_j}^t = \rho_{a_j}^{t-1+} \frac{\gamma_{min}}{|A-1|} \quad (11)$$

In fact, when a node at state s applies an action a and receives a higher reward, the probability of choosing the action a in the future increases. However, the node still keeps a small probability of choosing other actions, which allows it to continuously explore the dynamic environment preventing the node from getting stuck at local optima. On the other hand, if after applying an action a the node receives a smaller reward, the probability of selecting action a in the future is reduced by γ_{min}. Thus gradually more weight is given to more beneficial actions.

C. Smart-DTN-CC State Machine

Our approach entails that a node makes congestion control decisions based on its local information (that sometimes includes neighbor information) - the input rate, output rate and available buffer space at the node, which along with their derivatives are used to predict the level of congestion in a DTN. The possibility of congestion is indicated by a congestion detection daemon where a node can be in one of the states shown in Figure 1: Congested, Prospective-congested (PCongested), Decrease-congested (DCongested) and Non-congested (NCongested). Note that each node is able to predict if its buffer occupancy rate is increasing (EWMA-POSITIVE) or decreasing (EWMA-NEGATIVE). It does this using Exponentially Weighted Moving Average (EWMA) [33] as show in Equation 12, where Z_t is the buffer occupancy prediction at time t, B_{ocp} is the current buffer occupancy and Z_{t-1} is the buffer occupancy at time $t-1$.

$$Z_t = \alpha B_{ocp} + (1-\alpha) Z_{t-1} \quad (12)$$

In this context, the congestion level indicated by the daemon will depend on the available buffer space. In particular the node can make transition as shown in Figure 1 and as described above. Note that in Figure 1, *BUFFER OCP* is the current buffer occupancy rate and *BUFFER FREE* is the percentage of available buffer space.

1) Congested: the node makes a transition to *Congested* state if there is no available space on node's buffer

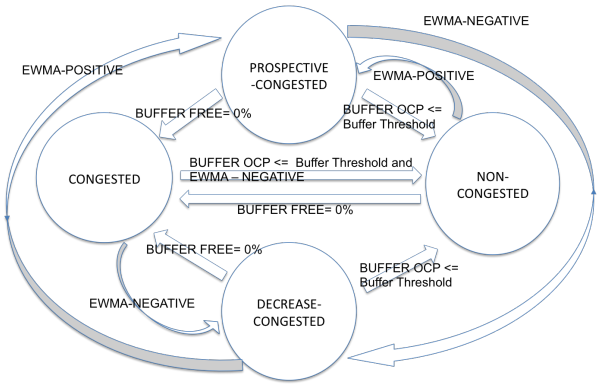


Fig. 1: Smart-DTN-CC state machine

2) PCongested: If the node's buffer occupancy rate is growing (*EWMA-POSITIVE*) it moves to *PCongested* state. This state indicates how imminent the threat of buffer exhaustion is.
3) DCongested: If the node's buffer occupancy rate is decreasing (*EWMA-NEGATIVE*) it moves to *DCongested* state. This state indicates that the node is already available to receive messages. Our approach uses this state to provide network utility.
4) NCongested: In this state the node is in a terminal state (the goal was reached by the controller). Our approach is conservative here in the sense that a buffer occupancy threshold (*Buffer Threshold*) exists that determines the transition for this state and at the same time the node's buffer occupancy rate must have a tendency to decrease (*EWMA-NEGATIVE*). This indicates the least and most likely of impending congestion the node is up to.

D. Actions

In this section, we present the actions that have been currently implemented in Smart-DTN-CC. They are summarized in Table I.

E. Reward Function

In reinforcement learning, the reward function maps each perceived state (or state-action pair) to a single number, the "reward", indicating the intrinsic desirability of that state. A reinforcement learning node's objective is to maximize the total reward it receives in the long run. Q-learning's reward function evaluates the results of an action in a given state in order to reward "correct" behavior by increasing the *Q-value* of good actions or by decreasing the *Q-value* of bad actions. Typically, the reward function is not defined in terms of the action taken but rather by the resulting state. In other words, the reward r is a function of the transition from s to s'. Additionally, rewards are associated with states and nodes are not just rewarded for arriving at a "good" state s but also continuously rewarded for remaining in state s.

Actions	States			
	Congested	Prospective-Congested	Decrease-Congested	Non-Congested
Increase message generation period	×	×		
Broadcast CN (Congestion Notification)	×	×		
Discard expired message	×	×	×	×
Discard old message	×	×	×	×
Discard random message	×	×		
Discard message that will expire before next contact arises	×	×		
Discard oldest messages until space available	×			
Migrate messages	×			
Broadcast DCN - Decrease Congestion Notification			×	
Decrease message generation period			×	
Receive messages		×	×	×
Forward messages	×	×	×	×

×: the action can be taken when the node is in the state

TABLE I: State action space table.

The rules dictated by the reward function impose certain behavior on the node when congestion takes place. Therefore, the function itself and its coefficients have to be carefully chosen in order to produce adequate results. As mentioned above, our approach associates the reward with the states. Table II shows the reward values we use for each state where reward values were chosen in the -1 to 1 range. Where -1 indicates that the node has made a transition to a bad state (Congested) and 1 indicates a transition to a good state (NCongested). In order to improve and to keep network utility high we associate a smaller positive reward to the transition to DCongested state of 0.5 and a smaller negative reward value to PCongested of -0.5.

III. Experimental Methodology

One of our main goals when evaluating Smart-DTN-CC is to show that it is able to deliver adequate performance in a wide range of DTN scenarios and conditions. To this end, we use the Opportunistic Network Environment (ONE) simulator [34], which is a simulation environment designed specifically for DTNs.

Each simulation ran for twelve simulated hours unless otherwise specified. Each data point is the average of at least 5 runs, with 95% confidence intervals displayed. Results when no congestion control is employed are used as performance baseline.

We use the following performance metrics in our study: (1) **delivery ratio** is the ratio between the number of received messages at destination nodes to the number of message originally transmitted by the source and (2) **end-to-end latency** is the average time to deliver messages to their destination. The parameters of the ONE simulator and the values we use in our simulations are listed in Table III. Simulation parameter values we used in our experiments were based on experiments reported in the DTN congestion control literature, in particular, in papers presenting the congestion control mechanisms we studied.

The scenarios we simulate include 50 nodes that move according to a pre-defined mobility regime. As they move, nodes encounter one another "opportunistically." During these opportunistic contacts nodes are able to exchange messages. Figure 2 shows the output of the ONE simulator's graphic interface illustrating a snapshot of a DTN scenario. We assume that all nodes have the same transmission range.

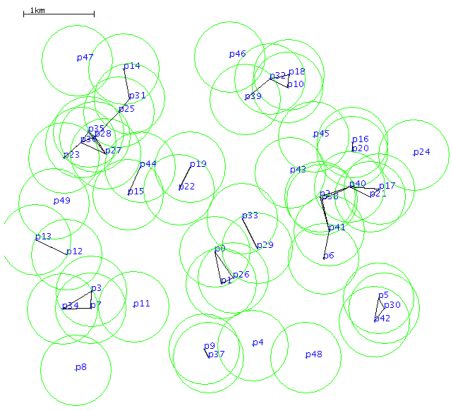


Fig. 2: DTN scenario.

Section IV-C presents results comparing the performance of Smart-DTN-CC against a set of congestion control mechanisms representative of the current DTN congestion control state-of-the-art, namely: AFNER [35], CCC [36], RRCC [37], and SR [38].

We evaluate Smart-DTN-CC when DTN nodes move according to three mobility regimes, namely: Random Walk (RW), Random Way Point (RWP), and Shortest Path Map-Based Movement (SPMBM). In RW [39], a node randomly chooses a destination within the simulation area. It then moves from its current location to the new one with speed uniformly distributed within the interval given by the *Group.speed* parameter. When it arrives at its destination, the node picks a new destination and repeats the steps above. The RWP mobility model [39] is a generalization of RW and works as follows: a mobile node picks a random destination within the simulated area; it then moves to that destination with constant speed chosen as a uniformly distributed random number in the interval *Group.speed*. When the node reaches its destination, it pauses for some time. In our simulations, the pause time is

States s	States s'			
	Congested	Non-Congested	Prospective-Congested	Decrease-Congested
Congested	-1	1	-	0.5
Non-Congested	-1	1	-0.5	-
Prospective-Congested	-1	1	-0.5	0.5
Decrease-Congested	-1	1	-0.5	0.5

- : the transition does not exist.

TABLE II: Reward function

Parameters		
Name	Description	Value
Scenario.endTime	simulation time	43200 seconds
btInterface.transmitSpeed	bandwidth	2.5 Mbps
btInterface.transmitRange	transmitting range	150 m
Group.router	routing protocol	[EpidemicRouter, ProphetRouter, SprayAndWaitRouter (10 msg copies)]
Group.movementModel	mobility model	[RandomWayPoint, RandomWalk, ShortestPathMapBased-Movement]
Group.bufferSize	node buffer size	4000 KB
Group.bufferThreshold	percentage value that indicates if the node is Non-Congested	[50, 60, 70, 80, 90]%
Group.alphaEWMA	the weight assigned to the current observation at EWMA function	[0.05, 0.20, 0.40, 0.60, 0.80]
Group.gammaGlearning	the discounted estimated future value at Q-learning function	[0.20, 0.40, 0.60, 0.80, 1.0]
Group.actionSelectionMethod	action selection methods	[boltzmann, wolf]
Group.gammaMin	minimum learning rate for WOLF action selection method	0.1
Group.gammaMax	maximum learning rate for WOLF action selection method	0.9 -$\rho_{\alpha_i}^{t-1}$
Group.msgTTL	message time to live	30000 seconds
Group.nrofHosts	number of nodes in network	50
Group.speed	max and min speed that the nodes must move	{0.5, 1.5}m/s
Movimentmodel.worldSize	area where simulation takes place	1 km$x$1 km (RandomWayPoint, RandomWalk) and 6 km$x$6 km (ShortestPathMapBasedMovement)
Events1.size	message size	{50, 100} KB
Events1.interval	Creation interval in seconds, i.e. one new message every 1 to 100 seconds	[1-100, 1-200, 1-300, 1-400, 1-500] seconds

TABLE III: Simulation parameters and their values

a uniformly distributed random number between $\{0, 120\}$ seconds. After that, the node picks another random destination and repeats the steps above. We note that we use the RWP mobility regime since it has been commonly used in the DTN literature, and specifically in the papers that propose the DTN congestion control schemes we use in our comparative performance evaluation of Smart-DTN-CC. The SPMBM [40] model uses Dijkstra's shortest path algorithm to calculate the shortest path from the current location to a randomly selected destination. Similarly to RWP, when the node arrives at its destination, it also uses a uniformly random pause time between $\{0, 120\}$ seconds.

Nodes generate messages according to a uniformly distributed random value within the interval specified by the $Events1.interval$ parameter. We vary the message generation rate according to the $Events1.interval$ intervals listed in Table III.

IV. Results

A. Cumulative Reward

One way to study the performance of reinforcement learning approaches is to evaluate the evolution of the cumulative reward [41] [42] over time. Figure 3 shows the average cumulative reward over all nodes for different mobility models as a function of simulation time. For these experiments, the values we use for the simulation parameters, which are shown in the caption of Figure 3, have been commonly used in the literature when evaluating Q-Learning approaches [41] [42].

In Smart-DTN-CC, the cumulative reward increases if the node makes a transition to either the NonCongested or DCongested states. Consequently, cumulative rewards that increase over time indicate satisfactory performance which is what we observe from Figure 3.

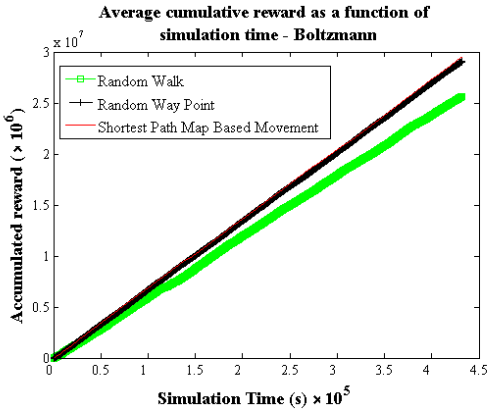


Fig. 3: Average cumulative reward as a function of the simulation time (Simulation Time 120 h, Buffer Size 4000 kB, Buffer Threshold 60%, α EWMA 0.80, γ Q-learning 0.2, Message Generation Period 300 s, Epidemic Routing, Boltzmann Action Selection Method).

B. Buffer Threshold Evaluation

Controlling buffer occupancy at network nodes is critical in any congestion control effort. As such, Smart-DTN-CC uses the buffer occupancy threshold to determine a node's current congestion level and trigger the corresponding state transition. The results presented in this section show the performance of Smart-DTN-CC as a function of the buffer threshold.

Figure 4 shows the average delivery ratio as a function of the buffer threshold for different routing protocols. These results were generated using RWP mobility because it is the mobility regime commonly used in the DTN congestion control literature. However, we observe similar behavior when we use both RW and SPMBM mobility. Our results show that the higher the buffer threshold, the more conservative the congestion control effort. In other words, as the buffer threshold increases, congestion control becomes less proactive and more reactive. This is consistent with what we observe from the graph in Figure 4 which shows an slight increase in delivery ratio as the buffer threshold increases up to a certain value, after which the delivery ratio decreases slightly. This behavior is observed for all routing mechanisms used.

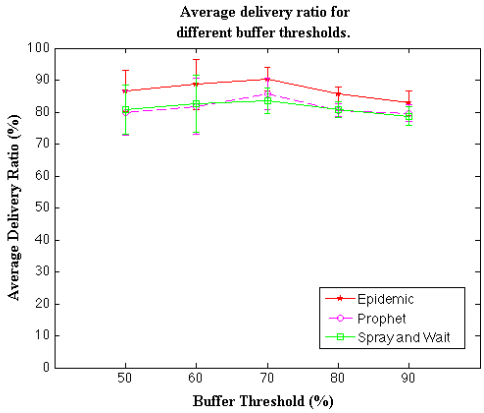


Fig. 4: Average delivery ratio as a function of the buffer threshold for different routing protocols (Simulation Time 12 h, Buffer Size 4000 kB, $\gamma_{max} = 0.9$, $\gamma_{min} = 0.1$, γ Q-learning 0.2, α EWMA 0.80, Random Way Point Mobility Model, Message Generation Period 300 s, WoLF Action Selection Method).

C. Comparative Performance Study

We also perform a comparative performance study of Smart-DTN-CC using as baseline a set of existing DTN congestion control mechanisms, namely AFNER [35], CCC [36], RRCC [37], and SR [38], which we implemented in the ONE simulator.

Figure 5 shows the average delivery ratio as a function of the message generation period. We observe that Smart-DTN-CC's average delivery ratio is consistently higher than all the other DTN congestion control mechanisms. We should also highlight that Smart-DTN-CC yields superior performance for both the Boltzmann and WoLF action selection methods.

Figure 6 shows Smart-DTN-CC's average delivery ratio compared to the different DTN congestion control mechanisms for different routing protocols and under different mobility models. Consistent with what was observed in Figure 5, the results show that Smart-DTN-CC using

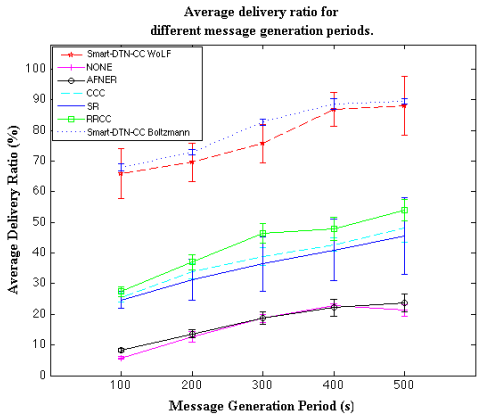


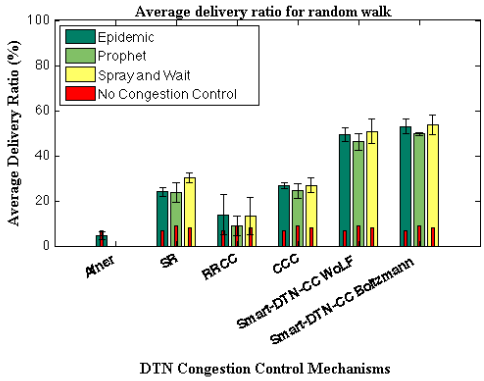
Fig. 5: Average delivery ratio as a function of message generation period (Buffer Size 1000 kB, Epidemic Routing, Random Way Point, Smart-DTN-CC (Buffer Threshold 60%, $\gamma_{max} = 0.9$, $\gamma_{min} = 0.1$, α EWMA 0.80, γ Q-learning 0.2)).

either WoLF or Boltzmann action selection methods outperforms the others mechanisms. Note that the same trend is observed for different routing protocols. This result highlights the fact that Smart-DTN-CC's behavior is independent of the underlying routing scheme, whereas some existing DTN congestion control mechanisms (e.g., AFNER and RRCC) were designed to operate over specific routing protocols.

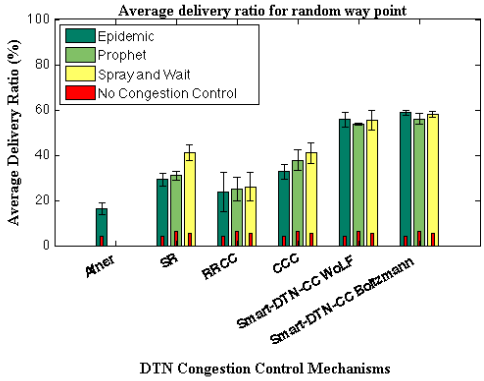
Figure 7 compares Smart-DTN-CC's average latency against the other DTN congestion control techniques under different mobility models. These results show that Smart-DTN-CC's latencies are significantly lower that the ones for all the other mechanisms under all the mobility regimes and routing protocols studied. Smart-DTN-CC's superior performance in terms of latency can be attributed to its hybrid approach to congestion control which combines proactivity and reactivity.

V. Conclusion

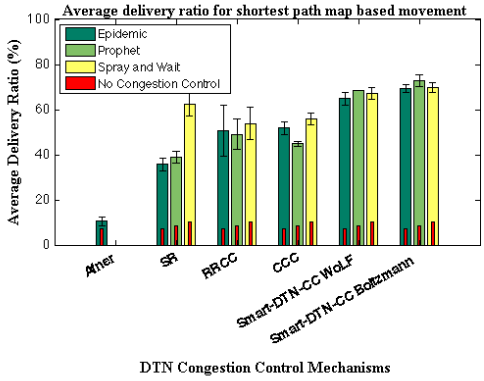
This paper introduced a novel congestion control framework for delay and disruption tolerant networks (DTNs) based on reinforcement learning. The proposed framework, named Smart-DTN-CC, adjusts its operation automatically as a function of the dynamics of the underlying network and requires no a-priori knowledge of the target environment. To our knowledge, Smart-DTN-CC is the first DTN congestion control framework that has the ability to automatically and continuously adapt to the dynamics of the target environment which allows Smart-DTN-CC to deliver adequate performance in a variety of DTN applications and scenarios. As demonstrated by our experimental evaluation, Smart-DTN-CC is able to consistently outperform existing DTN congestion control mechanisms under a wide range of network conditions and characteristics. As future work, we plan to test Smart-



(a) Random Walk

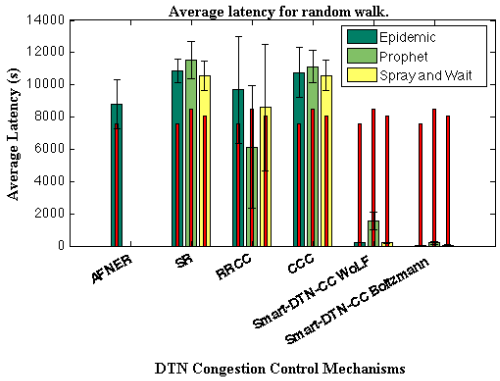


(b) Random Way Point

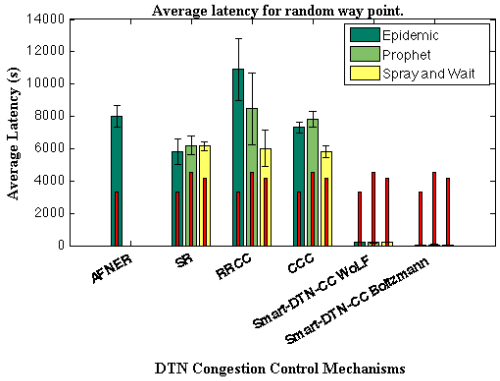


(c) Shortest Path Map Based Movement

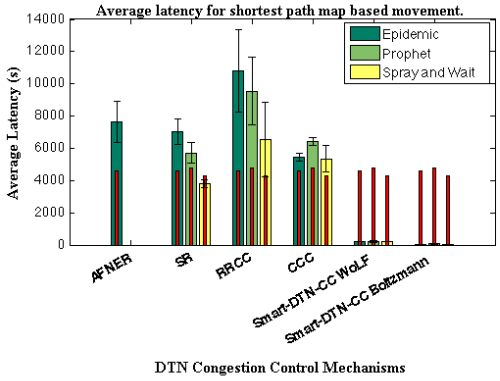
Fig. 6: Average delivery ratio for different mobility models and routing protocol (Buffer Size of 500 kB, Message Generation Period of 300 s, Buffer Threshold for Smart-DTN-CC of 60%).



(a) Random Walk



(b) Random Way Point



(c) Shortest Path Map Based Movement

Fig. 7: Average latency for different mobility models and routing protocols (Buffer Size of 500 kB, Message Generation Period of 300 s, Buffer Threshold for Smart-DTN-CC of 60%).

DTN-CC on a real testbed and evaluate its performance. In particular, we plan to use Smart-DTN-CC in deep space communication applications. To this end, besides simulations in interplanetary scenarios, we will also implement and test Smart-DTN-CC on the ION platform [43].

Acknowledgment

Some of the research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Government spon-

sorship acknowledged.

The Brazilian National Council for Scientific and Technological Development – CNPq (Process number 245492/2012-7) supported this project. And Coordenação de Aperfeiçoamento de Pessoal de nível Superior – CAPES (Process number BEX 5063/14-0) has also supported this project.

References

[1] K. Fall, W. Hong, S. Madden, and C. S. Madden, "Custody transfer for reliable delivery in delay tolerant networks," 2003.

[2] M.-C. Chuah, P. Yang, B. Davison, and L. Cheng, "Store-and-forward performance in a dtn," in *Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd*, vol. 1, May 2006, pp. 187–191.

[3] A. P. da Silva, S. Burleigh, C. M. Hirata, and K. Obraczka, "A survey on congestion control for delay and disruption tolerant networks," *Elsevier Ad Hoc Networks*, 2014.

[4] A. P. Silva, S. Burleigh, C. M. Hirata, and K. Obraczka, "Congestion control in disruption-tolerant networks: a comparative study for interplanetary networking applications," in *Proceedings of the 2014 MobiCom workshop on Challenged Networks - CHANTS'14*, 2014.

[5] ——, "Dtn congestion control unplugged: A comprehensive performance study," in *Proceedings of the 2015 MobiCom workshop on Challenged Networks - CHANTS'15*, 2015.

[6] S. Mahadevan and J. Connell, "Automatic programming of behavior-based robots using reinforcement learning." *Artif. Intell.*, vol. 55, no. 2, pp. 311–365, 1992. [Online]. Available: http://dblp.uni-trier.de/db/journals/ai/ai55.html#MahadevanC92

[7] V. Gullapalli, J. Franklin, and H. Benbrahim, "Acquiring robot skills via reinforcement learning," *Control Systems, IEEE*, vol. 14, no. 1, pp. 13–24, Feb 1994.

[8] S. Schaal, "Learning from demonstration," in *Advances in Neural Information Processing Systems 9*, M. Mozer, M. Jordan, and T. Petsche, Eds. MIT Press, 1997, pp. 1040–1046.

[9] A. Elwhishi, P. Ho, K. Naik, and B. Shihada, "ARBR: adaptive reinforcement-based routing for DTN," in *IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob 2010, Niagara Falls, Ontario, Canada, 11-13 October, 2010*, 2010, pp. 376–385. [Online]. Available: http://dx.doi.org/10.1109/WIMOB.2010.5645040

[10] L. Valerio, R. Bruno, and A. Passarella, "Adaptive data offloading in opportunistic networks through an actor-critic learning method," in *Proceedings of the 9th ACM MobiCom Workshop on Challenged Networks*, ser. CHANTS'14. New York, NY, USA: ACM, 2014, pp. 31–36.

[11] J. A. Bagnell and J. Schneider, "Autonomous helicopter control using reinforcement learning policy search methods," in *Proceedings of the International Conference on Robotics and Automation 2001*. IEEE, May 2001.

[12] U. Dogan, J. Edelbrunner, and I. Iossifidis, "Autonomous driving: A comparison of machine learning techniques by means of the prediction of lane change behavior," in *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, Dec 2011, pp. 1837–1843.

[13] "Predict the set of bird species present in an audio recording, collected in field conditions," 2013, oregon University.

[14] "Create an algorithm to detect north atlantic right whale calls from audio recordings, prevent collisions with shipping traffic," https://www.kaggle.com/c/whale-detection-challenge, 2013, marinexplore and Cornell University.

[15] R. J. Byrd, S. R. Steinhubl, J. Sun, S. Ebadollahi, and W. F. Stewart, "Automatic identification of heart failure diagnostic criteria, using text analysis of clinical notes from electronic health records," *International Journal of Medical Informatics*, vol. 83, no. 12, pp. 983 – 992, 2014.

[16] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: a survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[17] M. Mirza, "A machine learning approach to problems in computer network performance analysis," Ph.D. dissertation, University of Wisconsin-Madison, 2012.

[18] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Security and Privacy (SP), 2010 IEEE Symposium on*, May 2010, pp. 305–316.

[19] P. Geurts, I. E. Khayat, and G. Leduc, "A machine learning approach to improve congestion control over wireless computer networks," in *In Proc. of IEEE Int. Conf. on Data Mining (ICDM-2004*, 2004, pp. 383–386.

[20] W. Du, Y. Liao, N. Tao, P. Geurts, X. Fu, and G. Leduc, "Rating network paths for locality-aware overlay construction and routing," Nov 2013, http://orbi.ulg.ac.be/bitstream/2268/170423/1/rating_journal_final.pdf.

[21] M. A. Kaafar, S. Berkovsky, and B. Donnet, "On the potential of recommendation technologies for efficient content delivery networks," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 3, pp. 74–77, Jul. 2013.

[22] J. M. Francois, "Performing and making use of mobility prediction," Ph.D. dissertation, UNIVERSITE DE LI EGE Faculté des Sciences Appliquees Departement dâĂŹElectricité, Electronique et Informatique, 2007.

[23] B. Nunesa, K. Veenstra, W. Ballenthin, S. Lukin, and K. Obraczka, "A machine learning framework for tcp round-trip time estimation," *EURASIP Journal on Wireless Communications and Networking*, 2014.

[24] Y. Edalat, J. Ahn, and K. Obraczka, "Network state estimation using smart experts," in *11th International Conference in Mobile and Ubiquitous Systems (MOBIQUITOUS)*, 2014.

[25] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[26] R. S. Sutton, "Learning to predict by the methods of temporal differences," in *MACHINE LEARNING*. Kluwer Academic Publishers, 1988, pp. 9–44.

[27] C. Andalora, "Q-learning and the impact of exploration policies," http://www.cs.rit.edu/~cja9200/files/ArtificialIntelligence_ResearchPaper.pdf, November 2007.

[28] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, UK, May 1989. [Online]. Available: http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf

[29] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[30] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *In Proceedings of the Seventh International Conference on Machine Learning*. Morgan Kaufmann, 1990, pp. 216–224.

[31] M. Bowling and M. Veloso, "Rational and convergent learning in stochastic games," in *In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, 2001, pp. 1021–1026.

[32] J. Godoy, I. Karamouzas, S. J. Guy, and M. Gini, "Online learning for multi-agent local navigation," in *CAVE Workshop at AAMAS*, 2013.

[33] P. Zangari, "Estimating volatilities and correlations," New York: Morgan Guaranty, Tech. Rep., 1994.

[34] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. New York, NY, USA: ICST, 2009.

[35] L. Yun, C. Xinjian, L. Qilie, and Y. Xianohu, "A novel congestion control strategy in delay tolerant networks," in *Second International Conference on Future Networks*, 2010.

[36] L. Leela-amornsin and H. Esaki, "Heuristic congestion control for massage deletion in delay tolerant network," in *Smart Spaces and Next Generation Wired/Wireless Networking*, August 2010, third Conference on Smart Spaces and 10th International Conference.

[37] N. Thompson, S. C. Nelson, M. Baknt, T. Abdelzaher, and R. Kravets, "Retiring replicants: Congestion control for intermittently - connected networks," in *INFOCOM'2010, IEEE*, March 2010, pp. 1–9, university of Illinois at Urbana-Champaing, Department Of Computer Science.

[38] M. Seligman, K. Fall, and P. Mundur, "Alternative custodians for congestion in delay tolerant networks," in *SIGCOMM'06 Workshops*, 2006.

[39] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communications and Mobile Computing*, vol. 2, no. 5, pp. 483–502, 2002.

[40] A. Keränen and J. Ott, "Increasing reality for dtn protocol simulation," Helsinki University of technology, Networking Laboratory, Tech. Rep., July 2007.

[41] A. K. Yadav and S. K. Shrivastava, "Evaluation of reinforcement learning techniques," in *Proceedings of the First International Conference on Intelligent Interactive Technologies and Multimedia*, ser. IITM '10. New York, NY, USA: ACM, 2010, pp. 88–92.

[42] J. Insa-Cabrera, D. L.Dowe, and J. HernÃąndez-Orallo, "Evaluating a reinforcement learning algorithm with a general intelligence test." in *CAEPIA*, ser. Lecture Notes in Computer Science, J. A. Lozano, J. A. GÃąmez, and J. A. Moreno, Eds., vol. 7023. Springer, pp. 1–11.

[43] J. P. Laboratory, "ION: Interplanetary overlay network," https://ion.ocp.ohiou.edu/, 2013, accessed in January 2013.